

1 In this case, users wait an average of only 122.5 milliseconds instead of 242.5, obtained
2 using first-come, first served scheduling. More benefit is delivered sooner, and only less
3 valuable and more costly jobs are delayed. Additionally, a pre-emptive scheduler has the
4 ability, continuously, to insert higher priority jobs in front of lower priority ones. So, with the
5 above scheduling and pre-emptive scheduling, the final, 200-millisecond job may run later
6 than fourth, but it will not run sooner.

7
8 The invention can employs well-known techniques for optimization and job scheduling.
9 With the present invention known optimization and job scheduling techniques can be used
10 to provide efficient Web and Internet servers, independent of the particular optimization or
11 scheduling technique used
12

13 Classification can be done as a mathematical function of known and estimated parameters.
14 The above-mentioned benefit value may be a function of many parameters such as:
15 requested URL, client IP address, cookie, login, connection quality and other distinguishing
16 attributes. The cost may be more than just the time required to send or process the
17 request, including, such factors as the required bandwidth, CPU, latency, data generation
18 requirements, and total server load.

19
20 In general, the benefit and cost are functions of known and estimated parameters can be
21 described as follows:
22

23 $\text{Benefit} = B(p_1, p_2, p_3 \dots)$

24 $\text{Cost} = B(p_1, p_2, p_3 \dots)$, where p_N are known and estimated parameters.
25

26 The functions may be tabular, or an actual mathematical function of the parameters or a
27 combination of tabular and arithmetic functions. For example, a system can award points
28 for a desirable URL request, one that is know to encourage commerce, compared to a
29 reference or general information section of the site, which is read by both customers and
30 non-customers.

31
32 Order placement = 100 points

1 Browse catalog = 20 points

2 Download reference material = 10 points

3
4 Points also go to requests that correspond to requestors who are, for example, good
5 customers, as determined by the cookie, login, or, possibly, the IP source address.

6
7 Returning customer = +20 points

8
9 The connection quality determines how critical a connection is and how noticeable delays
10 will be. This allows re-sequencing non-critical requests behind critical ones. In some
11 cases, a modem user may not notice a slight delay but the DSL user will.

12
13 RTT estimate < 100 ms, critical, speed sensitive

14 RTT estimate > 100 ms, non-critical

15
16 The cost function is usually an estimate of the CPU required to generate the reply, the total
17 time including latency required to generate the reply and the bandwidth required to send
18 the reply back to the client. Other considerations include things such as the need for slots
19 on application and database servers.

20
21 For example, a typical response may cost 2 ms CPU, 20 ms latency, and 25k Bytes of data
22 transfer. In general the optimal scheduler is one that delivers the maximum benefit, subject
23 to the constraints that the total costs are less than the system limits in all cases.

24
25 Example Two: Server connection and load management: A special case of a managed
26 resource is a secondary server, usually an application server or database server. The
27 server may suffer performance problems if its load is too high or if there are too many
28 connections to the server from remote clients.

29
30 Typical server response time vs. load

31 1 pending requests -- 10 ms avg. response time

32 5 pending requests -- 12 ms avg. response time

1 10 pending requests – 20 ms avg. response time
2 20 pending requests -- 50 ms avg. response time
3 100 pending requests – 400 ms avg. response time
4

5 Here we see that, initially, efficiency increases due to increased concurrency and
6 overlapping of requests that have both a latency (delay) and a processing (CPU)
7 component. After a certain point, the server becomes less efficient due to overhead of
8 maintaining many pending requests. Many application and database servers use operating
9 system threads or processes to handle simultaneous tasks. This results in diminishing
10 returns as threads corresponding to pending requests compete for synchronization
11 primitives and as the operating system is forced to switch back and forth among the
12 outstanding tasks.
13

14 In the above example, the server runs most efficiently at a load of around 10 pending
15 requests, 20 ms average response time, for a total of around 500 "hits" per second. If the
16 load is 100, with an average response time of 400 ms, then the throughput is about 250 hits
17 per second. Intelligent load management would maintain the load on the server at 10,
18 while queuing the remaining requests. As described in Example 1, this queuing has the
19 added benefit of being able to order or prioritize the requests within the queue, with
20 additional gains in throughput and reduction of average response time.
21

22 With the present invention requests can be handled by an intermediate server/optimizer,
23 which queues the connection and transfers the data back and forth between the requesting
24 client and the origin and application servers.
25

26 Net result due to intelligent load management, with 100 pending requests:
27 500 hits/sec, with an average response time of $20 + (2\text{ms}) \cdot 90 = 200\text{ ms}$, compared with 250
28 hits/sec and 400 ms average response time with the standard application server.
29

30 Often, simply connecting to application and database servers slows the progress of tasks
31 executing on the server. In this case, it is helpful to off-load idle connections to an
32 intermediate server, which handles connection with an efficient queuing and I/O system.
33

A program flow diagram illustrating the operation of the system is given in Figure 2. First as indicated by block 201, the owner establishes a set of classifications priorities. These are stored in data base 123. If such a set are not as yet established the system utilizes a default set of priorities and classifications. The system receives requests from clients 100A to 100Z as indicated by block 203.

As indicated by block 295, the requests are classified in accordance with the classifications established by the system owner and stored in data base 123. Next as indicated by block 207, each request is prioritized in accordance with its classifications. Finally as indicated by block 208, the requests are scheduled in accordance with their priorities. Naturally higher priority requests are scheduled to be processes before lower priority requests.

Finally as indicated by block 209 the requests are sent to the web application server 110 and any outgoing traffic is sent to the network interface 120 for dispatch to the client machines 100A to 100Z.

Program 124 schedules the requests according to their priority and then prioritizes the requests in buffer 125. The requests are sent to the web application server 110 in accordance with their priority. A low priority requests which arrived first may reach web application server 112 after a high priority request which the system received at a later time.

In summary, the system includes programs that classify and priorities requests according to parameters established by the system operator. Different types of requests are provided with different priorities such that high priority requests are acted upon by the web application server 110 before low priority requests. This gives the system operator a great deal of flexibility in arranging the system provide a desired type of performance.

While in the embodiment described above, the classification, prioritization, and scheduling are done by three separate program routines, it should be understand, that the present invention relates to a program and system that performs this combination of functions. Those skilled in the art will realize that these functions can be performed using a wide variety of programming arrangements other than three separate programs.

1
2 In one embodiment of the present information, the classification, prioritization and
3 scheduling is done based upon the TCP payload data that is received at the system 101. It
4 is however noted that IP packets generally have the following form:

5 IP header gives source and destination information for routing.

6 TCP header: tells how many bytes in the payload, TCP options, etc.

7 TCP payload: Data bytes that contain a command such as that shown in Figure 3.

8
9 While one embodiment of the present invention operates by classifying, prioritizing and
10 scheduling requests based upon the payload data, alternate embodiments take into
11 account the information in the IP header and TCP header when the system does the
12 classification, prioritization and scheduling.

13
14 The above described embodiment of the invention merely improves performance by
15 scheduling the order in which tasks are operated upon by the server. In an alternate
16 embodiment, in addition to scheduling when the tasks will be provided to the server, the
17 amount of resources applied to each task is controlled in accordance with the
18 classification and priority of the task. For example, the amount of memory that the
19 server devotes to each class is controlled in accordance with the classification and
20 priority of the particular task. In such an embodiment, when each request is passed to
21 the server, a control parameter is also passed to the server. The control parameter
22 instructs the server concerning the amount of resources (for example memory) that
23 should be applied to the particular request.

24
25 The present invention optimizes CPU usage and network bandwidth while reducing
26 latency and providing feedback to server administrators. This is accomplished by
27 generating a cost-benefit model and optimizing it through request and task prioritization.
28 The invention can be implemented as a custom kernel with a lean thread model further
29 reduces requirements over a standard operating system's general-purpose thread
30 model. Essentially, the custom scheduler and resource allocator take control away from
31 the generic OS, returning it to the server owner. The custom kernel is fully event-driven,
32 responding quickly to common conditions such as data ready or disk I/O complete, as
33 well as to exception conditions like "connection reset by peer" or "address now

unreachable". The kernel layer is designed to conserve system resources, especially RAM, so that the server will function optimally and degrade gracefully. Most systems exhibit non-linear delay vs. load characteristics, with a sharp knee in the curve at a critical load, indicating non-graceful degradation. The present invention will extend the curve by deferring lower priority and "housekeeping" tasks and by using system resources more efficiently.

The cost-benefit model enables prioritization of requests by content (URL) or requester (IP address, login, cookie), as well as according to more automatic criteria such as content length, resource requirements, or end-to-end connection quality. For example, if the server has one large request and ten small requests it may wish to service the ten small requests first, satisfying ten users, while adding an acceptable delay to the large request. Furthermore, shaving 100 milliseconds off a 200 millisecond RTT (round trip time) task would result in noticeable increase in interactivity. However, shaving 100 milliseconds off a 600-millisecond modem connection would not even be appreciated. This targeted, fine-grain optimization is enabled by characterizing the requests, estimating their resource requirements, then queuing up the required tasks in the correct sequence to optimize the model.

A server in accordance with the present invention improves efficiency by performing resource allocation and scheduling according to a cost-benefit model that is established both automatically and by the server's administrator. Requests to the server may be classified according to URL, URL parameters, requestor, connection quality, content size and generation requirements, server required, time of day, or any other identifying characteristic.

Each class of request may have its own priority level, benefit, maximum or weighted proportional share of total bandwidth, maximum or weighted proportional share of an assigned CPU, or priority of access to any system resource, server, or storage device.

Each class may have deadlines or constraints on delivery, with variable penalties for lateness and dropping. External, user objectives and constraints are translatable to internal

constraints, which determine CPU and bandwidth scheduling and proportional share at the segment (packet) granularity.

With the present invention, a custom stack can schedules packet (segment) departure based on deadlines and lateness penalties that have been established by the scheduler and allocator

The invention utilizes an event-driven framework. A custom OS layer which manages the classification, prioritization and scheduling reduces the overhead of the general-purpose OS in the server, and it provides better communication between multiple, simultaneous tasks that are in progress. The custom OS layer maps multiple request/reply tasks to fewer threads or a single thread of the host OS.

The custom OS layer uses knowledge of continuations and non-blocking activities, co-operative multi-tasking based on a trust relationship between tasks is enabled. This is similar to the technique that is often used in the design of simulators. Such techniques reduce the overhead of multiprogramming a large number of independent tasks. Monitoring or a "pulse function" detects blocked or deadlocked processes to transfer workload to other, functioning processes, of which, new server OS instances may be added as needed.

A customized protocol stack can reduce the cost of open connections that have no assigned or discernible pending tasks. Such connections store only a source address and port without the usual socket resources. This stack may run in parallel with the existing TCP/IP stack by intercepting relevant ports, protocols, and URL requests at the kernel level, affording them special treatment.

The present invention eliminates network layer overhead. The custom stack also provides feedback to the scheduler and allocator regarding connection quality and available TCP and IP resources. Similarly, the custom stack greatly reduces the cost of servicing in-memory "cached" replies by forgoing the need for creating "socket" resources to grant access of kernel data to user spaces.

The custom stack may multiplex multiple connections and data transfers “on the wire” and at the network layer into a single user-level connection at the OS-user/application layer.

With the present invention, customized applications replace the layered network interface with interprocess communication (i.e. IPC) and remote procedure calls (i. e. RPC) to communicate directly with the system more efficiently.

One valuable by-product of peeking into the TCP layer to glean connection information is the ability to provide the server owner with more detailed traffic statistics. The server statistics and quality of achieving the user-defined and automatic criteria is fed-back to a monitoring and reporting application, which then displays said information to the administrator.

In conclusion: The present invention provides an event-driven custom kernel for a server. The custom kernel provides scheduling and resource allocation. The custom kernel operates in accordance with a cost-benefit model which is optimized by request and task prioritization. The cost-benefit model prioritizes requests by content (URL or requestor (IP address, login, or cookies) as well as according to criteria such as content length, resource requirements, or end-to-end connection quality. Tasks are classified and prioritized before being run on the CPU. Bandwidth is regulated and data departure is scheduled according to task and server specific criteria that can be established by a user. Fine-grain optimization is achieved by characterizing the requests, estimating their resource requirements, then queuing up the required tasks in the correct sequence to optimize the model. Several types of Interprocess communication (i.e. IPC) and remote procedure calls (i. e. RPC) are used to efficiently communicate directly with the system. These include providing feedback information between layers, and sending data directly from an internal layer to a receiver that is not an adjacent layer via inter-process communication. Since the kernel in the server obtains information from the TCP and IP layers, detailed traffic statistics can be provided to the server owner.

While the invention has been shown and described with respect to preferred embodiments thereof, it will be understood by those skilled in the art, the various changes in form and

1 detail can be made without departing from the spirit and scope of the present invention.
2 The invention is limited only by the appended claims.

3

4 I claim:

5

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000